# BetterMart

Group 138
Samridh Girdhar 2021282
Raunaque Khan 2021278

As Explained in Deadline 2, the database for BetterMart has a total of 10 entities:

1. Customer
2. Product
3. Product Feedback
4. Offer
5. Category
6. Orders
7. Cart
8. Delivery Partner
9. Payment
10. Retailer

## Schema Creation:

- In order to create a schema for our online retail store database, We first created a database on MySQL using the command:

  **CREATE DATABASE [database_name]**

- Then we inserted tables into the database for each of the above-stated entities, we did this using the syntax:

  **CREATE TABLE [database_name].[table_name](**

  **……**

  **);**

- And then using suitable Datatypes we added the attributes as mentioned in the ER diagram.

## Primary Keys:

A primary key is an attribute that acts as a unique Identifier.

We defined the primary keys using:

**PRIMARY KEY** (col_name)

Primary keys use **AUTO_INCREMENT** and are **NOT NULL**

| | Entity Name | Primary Key |
|---|---|---|
| 1 | Customer | Customer_ID |
| 2 | Product | product_ID |
| 3 | Product Feedback | product_ID |
| 4 | Offer | Offer_ID |
| 5 | Category | Category_ID |
| 6 | Cart | Customer_ID |
| 7 | Orders | Order_ID |
| 8 | Delivery Partner | DeliveryP_ID |
| 9 | Payment | Payment_ID |
| 10 | Retailer | Retailer_ID |

- Integrity Constraints:

  - Auto_Increment
  - Not Null
  - Primary Key
  - Foreign Key

- Indexes:

  Indexes are used to retrieve data from the database very fast. We created Indices using:

  **CREATE INDEX** idx_name **ON** table_name(col_name);

- DATA POPULATION:

  In order to produce bulk data (as in the case of customers, distributors, offers, etc.), we have used an online bulk data generator.

  The following syntax was used at some points:

  **INSERT INTO** database.tbl_name(col1,col2) **VALUES** (col1_val, 'col2_val'),

- Database :

  i) In the Back End, we have created a database and would be connected to the front end website through *MySQL*.
  ii) The Database is made such that it could easily handle large amounts of data and give concurrent results without performance degradation.
  iii) Throughout the database, atomicity has been assured as to reduce conflicts due to large amounts of entries.